Improving Batch Scheduling on Blue Gene/Q by Relaxing 5D Torus Network Allocation Constraints

Zhou Zhou, Xu Yang, Zhiling Lan
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616, USA
{*zzhou1, xyang56*}*@hawk.iit.edu, lan@iit.edu*

Paul Rich,* Wei Tang,† Vitali Morozov,* Narayan Desai,†‡
**Argonne Leadership Computing Facility*
†*Mathematics and Computer Science Division*
Argonne National Laboratory, Argonne, IL 60439, USA
‡*Ericsson Inc.*
200 Holger Way, San Jose, CA 95134, USA
**richp@alcf.anl.gov,* †*wtang@mcs.anl.gov*
**morozov@anl.gov,* ‡*narayan.desai@ericsson.com*

***Abstract*—As systems scale toward exascale, many resources will become increasingly constrained. While some of these resources have historically been explicitly allocated, many—such as network bandwidth, I/O bandwidth, or power—have not. As systems continue to evolve, we expect many such resources to become explicitly managed. This change will pose critical challenges to resource management and job scheduling. In this paper, we explore the potentiality of relaxing network allocation constraints for Blue Gene systems. Our objective is to improve the batch scheduling performance, where the partition-based interconnect architecture provides a unique opportunity to explicitly allocate network resources to jobs. This paper makes three major contributions. The first is substantial benchmarking of parallel applications, focusing on assessing application sensitivity to communication bandwidth at large scale. The second is two new scheduling schemes using relaxed network allocation and targeted at balancing individual job performance with overall system performance. The third is a comparative study of our scheduling schemes versus the existing one under different workloads, using job traces collected from the 48-rack Mira, an IBM Blue Gene/Q system at Argonne National Laboratory.**

I. Introduction

The demand for more computing power seems insatiable. Production systems already contain hundreds of thousands of processors, and they are headed to millions [1]. These systems are used to tackle scientific problems of increasing size and complexity, with diverse requirements for resources. The systems utilize shared resources, such as the communication infrastructure. to achieve higher performance while controlling costs. *As high-performance computing continues to evolve, these shared resources are becoming increasingly constrained*.

In order to harness the full potential of extreme-scale systems, *resource management* or *job scheduling* (i.e., effectively allocating available resources to applications) is of paramount importance. Modern resource managers focus primarily on effective use of job-dedicated resources such as processors, memory, and accelerators. Because of a variety of technology trends, the ratio of CPU and memory to shared resources is increasing. In the near future, *management of shared resources such as network and bandwidth will become increasingly critical*.

Torus-based networks are commonly used in high-end supercomputers because of their linear scaling on per-node cost as well as their competitive communication performance. The IBM Blue Gene/L, Blue Gene/P [2] [3], and Cray XT systems [4] use a 3D torus network for node communication. Blue Gene/Q (BG/Q) has its nodes electrically interconnected in a 5D torus network [5]. The K computer from Japan uses the "Tofu" system, which is a 6D mesh/torus topology [6]. On the recent Top500 list, 6 of the top 10 supercomputers use a high-radix torus-interconnected network [1]. In order to address the potential performance issues (e.g., job interference and communication contention) caused by shared torus networks, Blue Gene systems [2][3] use a *network partitioning* mechanism in which the network interconnect is reconfigured to provide private, per-job networks to compute nodes [7][]. Once a network partition is established, the job running on the partition can benefit from the dedicated synchronization network where all required hardware is dedicated to the job. Although a partition-based system provides jobs with dedicated network resources and bandwidth; however, the use of partitions introduces a new problem: resource contention caused by monotonically allocating shared network resources to a single job.

Such contention can cause unusable configurations of node resources, regardless of node state. For example, even if some nodes are idle, they still cannot be grouped together to serve a job because the wirings between them are occupied by other jobs. This issue can diminish both job response times and system utilization.

To address the problem, in this paper we investigate the potential of relaxing network resource allocation by utilizing application communication features. The partition-based design in Blue Gene systems provides a unique opportunity to explicitly allocate network resource (i.e., links or bandwidth) to jobs in a way that is impossible on other systems. While

this capability is currently rare, we expect it to become more common. More specifically, this paper makes three major contributions:

1) Substantial benchmarking of applications, focusing on assessing application sensitivity to network configuration at large scale. In particular, we evaluate a number of parallel benchmarks and DOE leadership applications on the production 48-rack Blue Gene/Q system Mira at Argonne by analyzing their performance variation under different network configurations.
2) Design of two new scheduling schemes for Mira. These two schemes have different characteristics, including network configurations and scheduling policies. In particular, we propose a communication-aware scheduling policy that selectively allocates network resource to users' jobs according to job communication characteristics.
3) Comprehensive evaluation of our scheduling schemes versus the current one used on Mira, through trace-based simulations using real workloads from Mira.

We have made the following major findings:

1) Not all applications are sensitive to the network bandwidth variation. Application performance under different network configurations depends mainly on their communication patterns and the proportion of communication over the total application runtime.
2) Our new scheduling schemes can significantly improve scheduling performance by up to 60% in job response time and 17% in system utilization.

The remainder of this paper is organized as follows. Section II introduces background about the Blue Gene/Q system Mira at Argonne. Section III presents the results of benchmarking applications on Mira. Section IV presents our new batch scheduling schemes. Section V presents a scheduling study. Section VI discusses related work. Section VII summarizes our conclusions and points out future work.

II. Background

A. Mira: The IBM Blue Gene/Q at Argonne

Mira is a 10 PFLOPS (peak) Blue Gene/Q system operated by Argonne National Laboratory for the U.S. Department of Energy [1]. It is a 48-rack system, arranged in three rows of sixteen racks. Each rack contains $1{,}024$ sixteen-core nodes, for a total of $16{,}384$ cores per rack. Mira has a hierarchical structure: nodes are grouped into midplanes, each midplane contains 512 nodes in a $4 \times 4 \times 4 \times 4 \times 2$ structure, and each rack has two such midplanes. Each node has 16 cores, giving a total of $786{,}432$ cores. Mira was ranked fifth in the latest Top500 list [1]. Mira uses a 5D torus-connected network. Each node in the machine has a unique set of coordinates on the full machine partition. Mira is a capability system, with single jobs frequently occupying substantial fractions of the system. The smallest production job on Mira occupies 512 nodes; 8,192-node and 16,384-node jobs are common on the system; larger jobs also occur frequently. Jobs up to the full size of Mira run without administrator assistance. Time on Mira is awarded primarily through the Innovative and Novel Computational Impact on theory and Experiment (INCITE) program [8] and the ASCR Leadership Computing Challenge (ALCC) program [9].

B. Partitioning on Mira

Mira uses network partitioning for job scheduling, and partitions can be constructed only in a limited set of ways. A partition must be a uniform length in each of the dimensions. Midplanes used to build partitions must be connected through a single dimension and form a roughly rectangular prism in five dimensions. Because the final dimension is used only to connect nodes within a single midplane, all partitions are length 1 in the *E* dimension. Additionally, the creation of a partition uses network resources in a dedicated fashion, preventing their use in other partitions. For a partition size to be valid, there must be a set of partition lengths in each dimension that results in a properly sized 5D prism of nodes. Partitions also require use of complete midplanes, so all partitions on the system are multiples of 512 nodes. For a given size, several partition variants may exist with different shapes.

Figure 1 illustrates the flat view of the network topology of Mira with two halves and three rows. Each square labeled with "RXX" represents a rack. Each rack contains two vertical midplanes (not shown in the figure). As we can see, the whole machine is split into six 8-rack sections. Each node in Mira has a unique logical coordinate (A, B, C, D, E). Given the logical coordinate of a node, we can translate the logical address of a node to the midplane location. The *A* coordinate decides which half of the machine the node is on. The *B* coordinate decides which row of the machine the node is on. The *C* coordinate refers to a set of four midplanes in two neighboring racks. The blue lines represent the cables that link racks together. Since the coordinate is based on a logical location and follows the midplane cable, this coordinate appears to jump around the 8-rack segment as illustrated in the figure. The *D* coordinate refers to a single midplane in two neighboring racks. Since the cable makes a loop around two racks, the coordinate loops in a clockwise direction.

C. Network Resource Contention

One unique feature of the Blue Gene/Q architecture is the ability to explicitly allocate network performance to jobs. When building a partition, a shared pool of network resources is allocated to a single partition at a time. If sufficient resources are dedicated to a partition, it will have a torus network. Alternatively, if fewer resources are allocated, the partition will have only a mesh network, in which the outside faces of the mesh are not connected except where

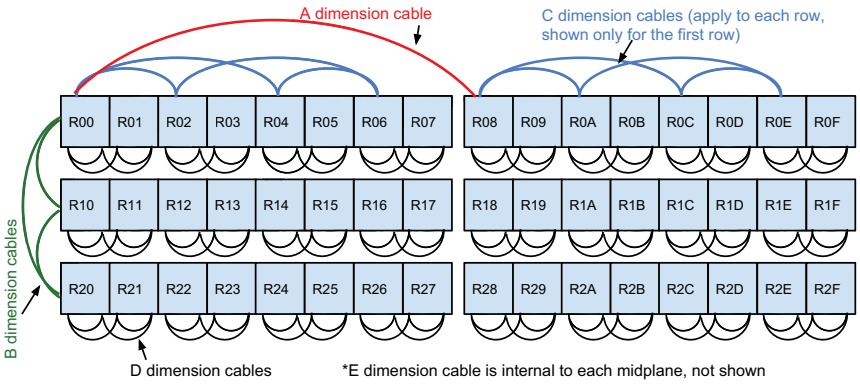
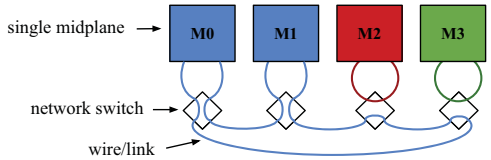
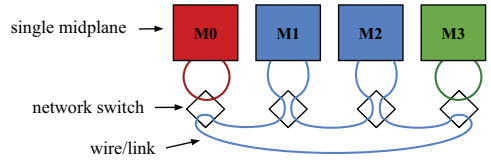


Figure 1. Flat view of Mira's network topology



(a) M0 and M1 as a partition



(b) M1 and M2 as a partition

Figure 2. Wire contention between midplanes. The figure is a schematic representation of a four-midplane-long dimension showing a two-midplane torus and two single midplane tori. Because of the wiring of the 2-midplane torus (blue) and the exclusivity of partitioning on Mira, this wiring prevents the formation of a torus or a mesh with the remaining two midplanes in this dimension. As shown, once two midplanes (M0 and M1 in (a), M1 and M2 in (b)) are linked together to form a 1K partition, they will consume all the wire resources along this four-midplane-long dimension. This is representative of both the C and D dimensions on Mira.

the internal midplane faces connect to one another. The performance of the torus partition is considerably better than that of the mesh; the worst case and average hop counts between nodes are reduced, and more bandwidth is available to applications. This difference in performance will affect application performance, particularly for communication-intensive applications.

Network contention is a substantial challenge for resource allocation on Blue Gene/Q systems. For example, it is possible that idle midplanes cannot be wired together to satisfy a job's resource request, because of a lack of wiring resources, as shown in Figure 2. This situation can occur even when midplane positions meet all geometric constraints for partition creation.

D. Current Job Scheduling on Mira

Batch scheduling on a partition-based system comprises two parts: ***network configuration*** and ***scheduling policy***. For network configuration, Mira uses a configuration in which all partitions are fully torus-connected. The partition size ranges from 512 nodes (single midplane) to the whole machine (48 racks). The scheduling policy has two phases. First, the resource manager uses a policy called WFP to order the jobs in the queue [10] [11] [12]. WFP favors large and old jobs, adjusting their priorities based on the ratio of their wait times to their requested runtimes. Upon each scheduling, the job at the head of the wait queue is selected and allocated to a partition. Then, a least-blocking (LB) scheme is used to choose the partition that causes the minimum network contention out of all candidates [11].

III. Application Benchmarking

We first investigate the impact of partition configuration on application performance. We choose four parallel benchmarks and three DOE leadership applications for benchmarking. To quantify performance difference, we define

$$runtime\_slowdown = \frac{T_{mesh} - T_{torus}}{T_{torus}}, \qquad (1)$$

where T_{mesh} is the application runtime on a mesh partition and T_{torus} is the application runtime on a torus partition.

A. Parallel Benchmarks and Applications

In this paper, we use the NAS Parallel Benchmarks, in particular NPB3, which has a larger problem size (class E) [13]. We choose three kernel benchmarks: LU, FT, and MG. LU solves synthetic systems of nonlinear partial differential equations. FT solves a three-dimensional partial differential equation using a fast Fourier transform (FFT). MG solves a three-dimensional discrete Poisson equation using the V-cycle multigrid method.

We also study four scientific applications: Nek5000, FLASH, DNS3D, and LAMMPS. The applications are used routinely by a number of INCITE projects.

Nek5000 [14] is a spectral element CFD code developed at Argonne National Laboratory. It features spectral element multigrid solvers coupled with a highly scalable, parallel coarse-grid solver. It was recognized in 1999 with a Gordon Bell prize and is used by more than two dozen research institutions worldwide for projects including ocean current modeling, thermal hydraulics of reactor cores, and spatiotemporal chaos.

FLASH 4.0 [15] is the lastest FLASH release from the ASC Center at the University of Chicago. The FLASH code [16] is a multiphysics simulation code written in Fortran90 and C using MPI with OpenMP. The driven turbulence setup is run using the split-PPM hydrodynamics solver and the uniform grid module, in a weak-scaling mode. This problem was run at a large scale on the BG/L at Lawrence Livermore National Laboratory [17] and is known to be highly scalable.

DNS3D is a direct numerical simulation code that solves viscous fluid dynamics equations in a periodic rectangular 3-D domain with a pseudo-spectral method of fourth-order finite differences and with the standard Runge-Kutta fourth-order time-stepping scheme [18]. DNS3D is highly dependent on network performance, since during each time step it executes three Fourier transforms for three 3-D scalar variables. This approach can effectively be transformed into all-to-all–type computations.

LAMMPS ("Large-scale Atomic/Molecular Massively Parallel Simulator") [19] is a general-purpose molecular dynamics software package for massively parallel computers. Developed at Sandia National Laboratories, it is written in an exceptionally clean style that makes it one of the most popular codes for users to extend, and it currently has dozens of user-developed extensions.

B. Benchmarking Results

Each of these benchmarks and applications was executed on the partitions of sizes 2K, 4K, and 8K nodes, using both torus and mesh partitions.

Table I
APPLICATION RUNTIME SLOWDOWN

Name	Runtime Slowdown		
	2K	4K	8K
NPB:LU	3.25%	0.01%	0.03%
NPB:FT	22.44%	23.26%	21.69%
NPB:MG	0.00%	11.61%	19.77%
Nek5000	0.95%	0.02%	0.44%
FLASH	0.83%	5.48%	4.89%
DNS3D	39.10%	34.51%	31.29%
LAMMPS	0.02%	0.87%	0.97%

Table I presents application slowdowns of NPB benchmarks. Obviously, LU is not sensitive to the switching from torus to mesh. It appears to have less than 4% slowdown at size 2K and close to zero slowdown when the computing scale is increased to 4K and 8K. The algorithm of LU is not highly parallelized, and most of its MPI routines are blocking communication. This leads to no performance loss when the network topology configuration is changed from torus to mesh.

MG shows no slowdown at size 2K. When the computing scale is increased, however, we observe a 12% slowdown at size 4K and nearly 20% slowdown at size 8K. MG has unique communication patterns. In particular, it involves both near-neighbor communication and long-distance communication, so its performance is sensitive to network topology changes.

FT also is sensitive to network topology. At all three sizes, its slowdown is more than 20%. The code performs global data communication for its FFTs [20]. This is the main reason that the performance drops significantly when using mesh partitions with reduced bisection bandwidth.

The slowdown results of the leadership applications also are presented in Table I. For LAMMPS and Nek5000, the use of mesh partitions has minimal impact on their performance: the slowdowns are always less than 1%.

In Nek5000, every process is communicating to 50 to 300 geometrically neighbor processes, which in practice means about 2 to 3 hops away from the source. For a torus, the process on the "border" node does not notice any difference because, in some sense, there are no borders in torus topology. For a mesh, the process will have half of the neighbors located in the same semi-plane, as in torus partition, but half the others will need to reuse the path of the semi-plane. The slowdown really depends on the level of mutligrid refinement and the placement of the processes relative to each other.

In FLASH, the slowdown is no more than 5% on the 4K and 8K partition. FLASH's runtime is dominated by computation, not communication, and at the larger sizes takes up on the order of 14% of the runtime. The reason is that the communication algorithm is largely point to point and generally fairly local. Because of the periodic boundary conditions of the physics in the problem, we do get a small but significant amount of off-node communication on the wraparound links. For example, for 8K partitions, the torus spent only 14% of its time in communication, whereas the mesh partition had communication for 17% of the runtime. We saw 23% slowdown in communication, which was translated into about 5% slowdown of runtime.

DNS3D exhibits substantial slowdown when switching from a torus to a mesh partition. Among the three partition sizes from 2K to 8K, the slowdown is always above 30%. In some cases (e.g., 2K), the slowdown is close to 40%. Our MPI profiling shows that DNS3D spends 60% of its runtime in MPI_Alltoall(). MPI_Alltoall() is scaling proportional to the bisection bandwidth of a partition. If one of the partition dimensions becomes a mesh, the bisection bandwidth of the partition is reduced by half. Therefore, it takes two times longer for MPI_Alltoall() to complete. That is why we observe 30% performance degradation here.

Clearly, certain applications are sensitive to communication bandwidth, especially those heavily using MPI collective calls.

Our benchmarking results demonstrate that application communication pattern is a key factor influencing application runtime under different network configurations. Applications dominated by local communications are not sensitive to the network topology changing from torus to mesh, whereas applications having a substantial amount of long-distance or global communications are prone to performance loss when running on mesh partitions.

IV. New Batch Scheduling

In this section we present two new scheduling schemes to improve batch scheduling. While this work targets Mira, these new scheduling designs are applicable to all Blue Gene/Q systems and other 5D torus-connected systems.

A. Contention-Free Partition

Based on the Mira's flexible controlling of switches and wiring, we build a new partition configured with mixed torus and mesh dimensions. In this paper, we call it "contention-free" partition. This type of partition has some dimensions as torus-connected with wrap-around links and other dimensions as mesh-connected. By building such partitions, we can ensure that the wiring contention does not happen uniformly on all dimensions. For example, we turn the *D* dimension of 1K partition into mesh, while still having the other four dimensions torus-connected. This contention-free 1K partition does not consume any extra wiring resources compared with a mesh partition, and it can provide better communication performance. Similar to a mesh partition, this new 1K partition also does not cause any wiring contention on its torus-connected dimension. On Mira, we build such partitions at sizes of 1K, 4K, and 32K. Compared with full mesh partitions, these contention-free partitions cause less performance degradation on application runtime. This is due to the fact that an application can still benefit from the torus links on dimensions A, B, C, and E.

B. Scheduling Using Relaxed Allocation Constraints

Using mesh and contention-free partitions requires fewer links than full torus partitions. According to the benchmarking results presented in Section III, we can observe that for a majority of applications, the performance loss caused by mesh configuration is not substantial. Hence, we propose two scheduling schemes using such partitions to relax resource allocation constraints and improve system scheduling performance.

1) MeshSched: We propose a mesh-based scheduling policy that uses a full mesh network configuration. This configuration is generated from the current one on Mira by turning every torus partition into a mesh partition except the 512-node partition, which must be a torus. This means wrap-around torus links are turned off in each dimension, consequently reducing the potential link contention between neighboring partitions. Resources can be more freely allocated without the constraint of wrap-around links. Obviously, runtime slowdown may occur for some communication-intensive applications since mesh partitions reduce the bisection bandwidth between two nodes.

2) Contention-free and Communication-aware (CFCA) : We also propose a new scheduling scheme using contention-free partitions and a scheduling policy that takes an application's communication intensity into account. Compared with mesh partitions, the new contention-free partitions can preserve the application performance as much as possible without causing resource contention. We built a new network configuration on Mira by adding these contention-free partitions to the current configuration on Mira. We also develop a communication-aware scheduling policy as shown in Figure 3. The new scheduling scheme allocates communication-sensitive jobs to torus partitions and allocates non-communication-sensitive jobs to contention-free partitions. By doing so, this scheduler seeks to balance user requirements and system performance. Note that the single 512-node midplane must always be a torus. Any jobs requiring no more than 512 nodes should directly be routed to a single midplane. Further, with the performance monitoring support on Mira, an application's sensitivity to network topology can be determined empirically.

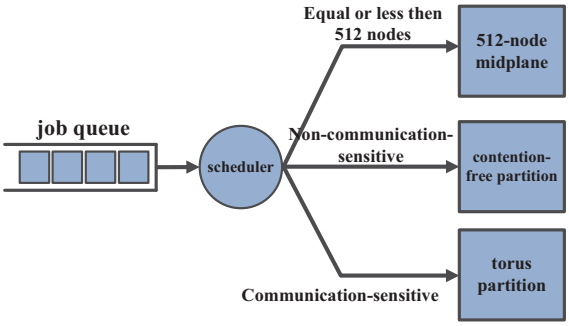


Figure 3. Communication-aware job scheduling using contention-free partitions

V. Experiments

In this section, we compare our new scheduling methods under a variety of workloads using trace-based simulation. The goal is to investigate the benefit of our design compared with the current one used on Mira.

A. QSim Simulator

Qsim is an event-driven scheduling simulator for Cobalt, the resource management and job scheduling package used on the 48-rack Mira. Taking the historical job trace as input, Qsim quickly replays the job scheduling and resource allocation behavior and generates a new sequence of scheduling

events as an output log. Qsim uses the same scheduling and resource allocation code that is used by Cobalt and thus will provide accurate resource management and scheduling simulation. Qsim is open source and available along with the Cobalt code releases [10]. It was used in our previous work [11][21][7].

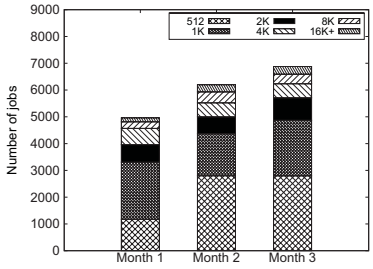


Figure 4. Job size distribution

B. Job Trace

We use a three-month workload trace collected from Mira. Figure 4 summarizes the jobs in these months. As we can see, the 512-node, 1K, and 4K jobs are the majority. For months 2 and 3, 512-node jobs account for half of the jobs. While the number of large-sized jobs (more than 8K nodes) is relatively low, these jobs consume a considerable amount of node-hours because of their sizes.

C. Evaluation Metrics

Four metrics are used for scheduling evaluation:

- *Average job wait time*. This metric denotes the average time elapsed between the moment a job is submitted and the moment it is allocated to run. It is commonly used to reflect the "efficiency" of a scheduling policy.
- *Average response time*. This metric denotes the average time elapsed between the moment a job is submitted and the moment it is completed. Similar to the above metric, it is often used to measure scheduling performance from the user's perspective.
- *System utilization*. System utilization rate is measured by the ratio of busy node-hours to the total node-hours during a given period of time [22] [23]. The utilization rate at the stabilized system status (excluding warm-up and cool-down phases of a workload) is an important metric of how well a system is utilized.
- *Loss of capacity (LoC)* This metric measures system fragmentation [7]. A system incurs LoC when it has jobs waiting in the queue to execute and when it has sufficient idle nodes but still cannot execute those waiting jobs because of fragmentation. A scheduling event takes place whenever a new job arrives or an executing job terminates. Let us assume the system has N nodes and m scheduling events, which occur when a new job arrives or a running job terminates, indicated by monotonically nondecreasing times t_i, for $i = 1...m$. Let n_i be the number of nodes left idle between the scheduling event i and $i+1$. Let δ_i be 1 if any jobs are waiting in the queue after scheduling event i and at least one is smaller than the number of idle nodes n_i and 0 otherwise. Then LoC is defined as follows.

$$LoC = \frac{\sum_{i=1}^{m-1} n_i(t_{i+1} - t_i)\delta_i}{N \times (t_m - t_1)} \quad (2)$$

Table II
SCHEDULING SCHEMES USED IN THE EXPERIMENTS

Name	Network Configuration	Scheduling Policy
Mira	Current config used on Mira	WFP and LB (see Section II)
MeshSched	All possible mesh partitions and 512-node torus	WFP and LB
CFCA	Current config used on Mira and additional contention-free partitions (1K, 2K, and 32K)	Communication-aware policy described in Figure 3

D. Results

In our experiments, we compare our new scheduling methods (termed *MeshSched* and *CFCA*) with the one currently used on Mira. Table II summarizes these scheduling methods.

We categorize jobs into communication-sensitive and non-communication-sensitive jobs. For each simulation, we set five slowdown levels for applications running on mesh partitions: 10%, 20%, 30%, 40%, and 50%. We also tune the percentage of communication-sensitive jobs in the workload. Similarly, five ratios are used: 10%, 20%, 30%, 40%, and 50%. We conduct experiments using the workload on a monthly base (3 months). In total we have 225 ($3 \times 3 \times 5 \times 5$) sets of experiments.

Because of space limitations, we present only a few representative results. To improve the figure readability, we present the results when the percentage of communication-sensitive jobs is 10%, 30%, or 50%. For system utilization, we present the relative improvement of *MeshSched* and *CFCA* over *Mira*.

Figure 5 shows the scheduling performance when runtime slowdown is set to 10%. First, we observe that both the *MeshSched* and *CFCA* schemes can have a striking effect on job wait times and response times for all three months. The largest wait time reduction is more than 50% for month 1 when there are 10% communication-sensitive jobs. The response time is also reduced substantially because of the reduction in job wait time. The relative improvement in job response time is smaller than that achieved in job wait time. It indicates that for most jobs their runtimes dominate the total response time. Second, we notice that *MeshSched* outperforms *CFCA* regarding wait time and response time for month 1. With a relatively low slowdown of 10%,

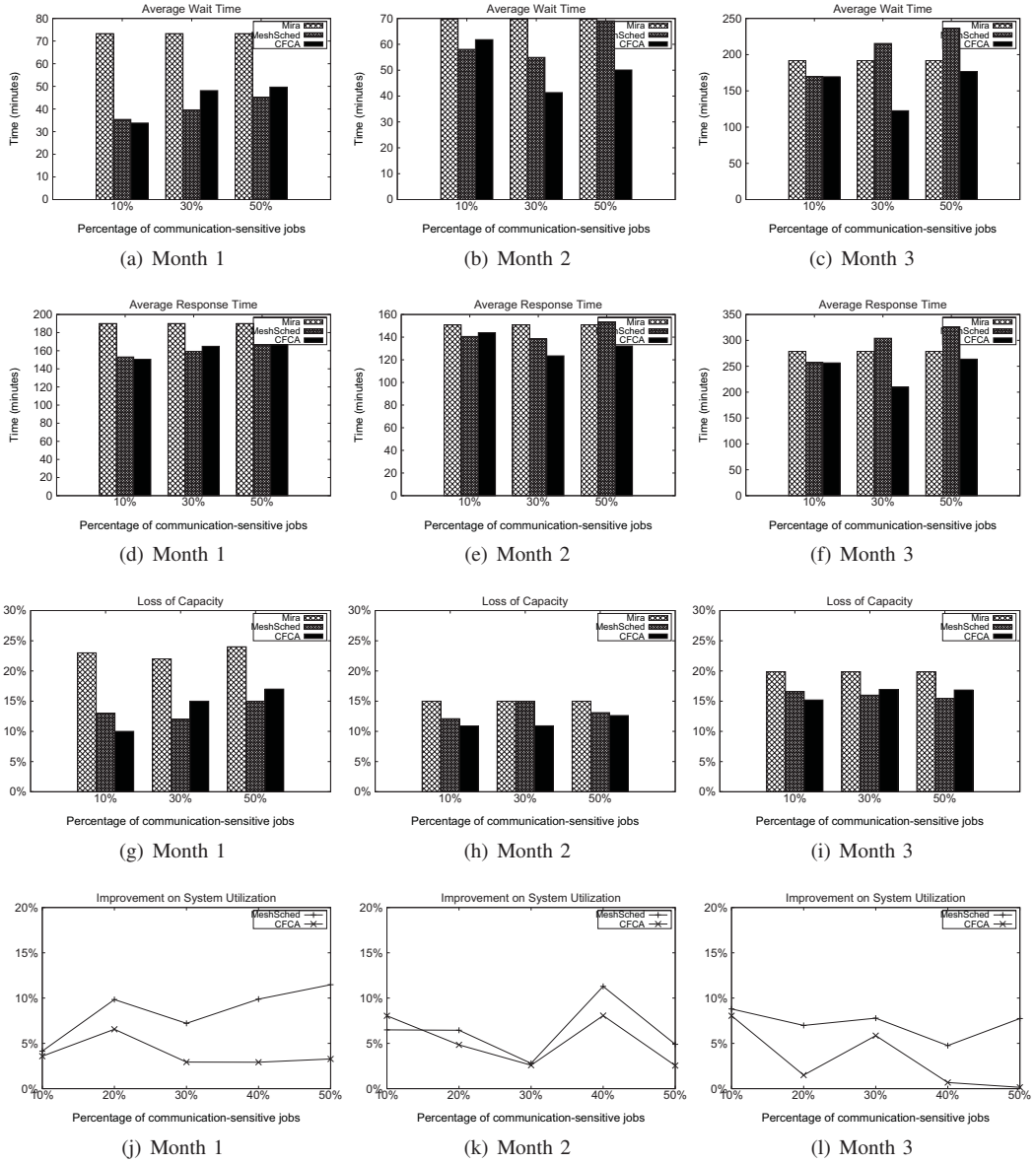


Figure 5. Comparison of scheduling performance using different scheduling policies, where runtime slowdown is set to 10% for communication-sensitive jobs

using mesh partitions provides shorter turnaround time with affordable performance loss. Third, with respect to LoC, both *MeshSched* and *CFCA* perform better than *Mira*. For month 1, LoC decreases more than 10% when there are 10% communication-sensitive jobs. This decrease is significant when we consider the machine scale. For Mira in a single month, approximately $2{,}538{,}944\ (= 0.1 \times 30 \times 24 \times 49152)$ node-hours are saved, enabling the system to run 72 hours at full load. *MeshSched* reduces more LoC than *CFCA* does. The reason is that *MeshSched* contains only mesh partitions except for 512 nodes, whereas *CFCA* still uses some torus partitions, inevitably causing more resource contention than does *MeshSched*. Clearly, both *MeshSched* and *CFCA* improve the overall system utilization. *MeshSched* can improve the utilization by more than 10% in month 2 with 40% communication-sensitive jobs. Although *CFCA* does not improve the utilization as much as the *MeshSched* does, the average improvement is about 5%, with the biggest improvement in month 3 when there are 10% communication-sensitive jobs. The reason is similar to the case of LoC; that is, *MeshSched* has much less network resource contention.

Figure 6 presents scheduling performance when runtime slowdown is set to 40%. With respect to job wait time, the *CFCA* scheme always outperforms the other two scheduling policies. For example, in month 1 with 10% communication-

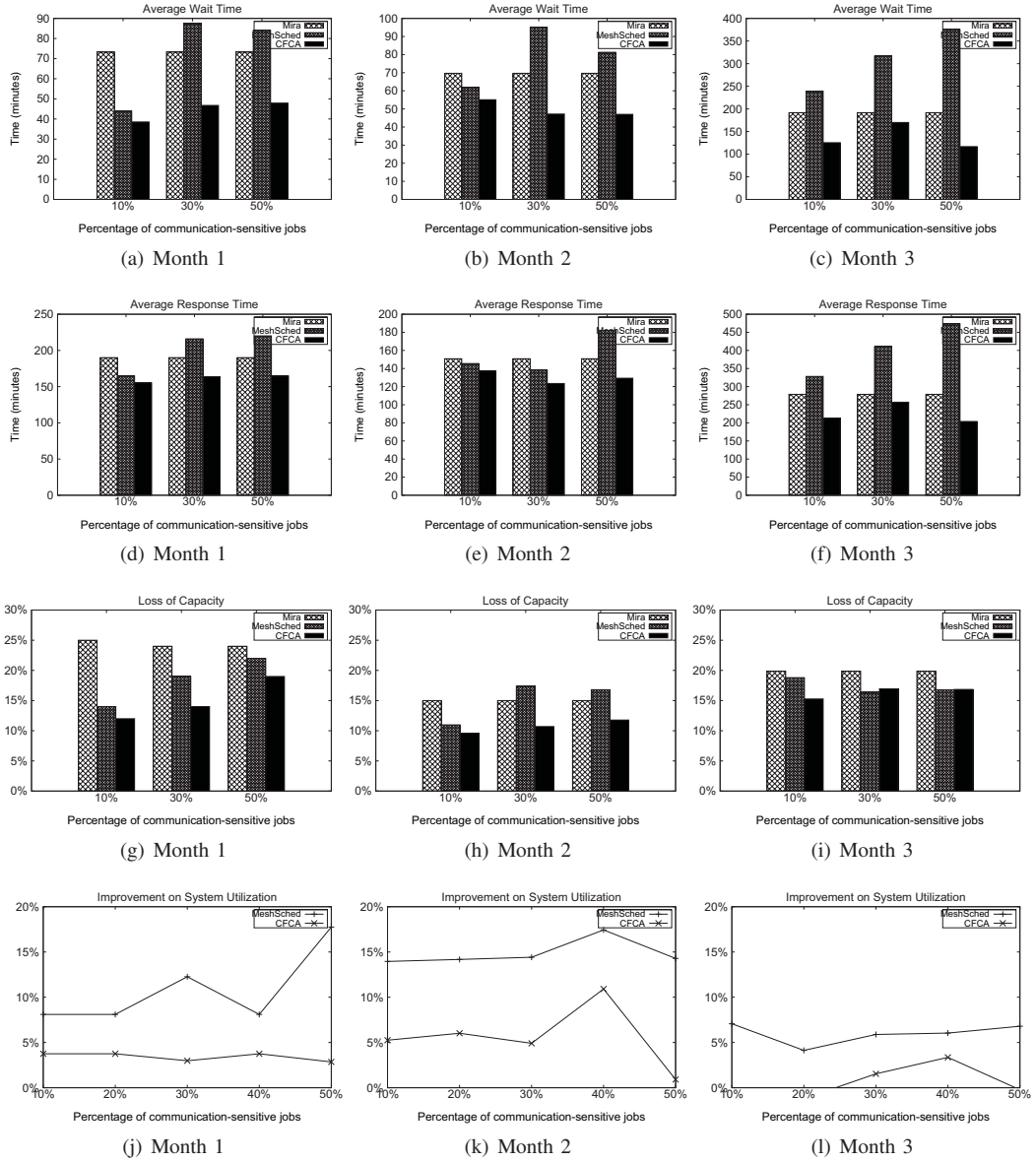


Figure 6. Comparison of scheduling performance using different scheduling policies, where runtime slowdown is set to 40% for communication-sensitive jobs

sensitive jobs, *CFCA* reduces the wait time by more than 50%. Similar performance improvement is achieved for job response time by using *CFCA*. As in the previous case, *MeshSched* generally results in a worse job performance than *Mira* does when there are more than 10% communication-sensitive jobs. In months 2 and 3, the job wait time is increased by 100%. The reason is that although the resource contention is reduced by using *MeshSched*, user jobs suffer from the substantial runtime expansion caused by using mesh partitions. Similar to Figure 5, both *MeshSched* and *CFCA* improve LoC. Especially in month 1, *CFCA* can greatly reduce LoC, much more than that achieved by using *MeshSched*. With respect to system utilization, *MeshSched* achieves more than 15% increase in some cases. Similar to Figure 5, *MeshSched* improves utilization more than *CFCA* does.

In summary, our main observations are as follows:

- *CFCA* outperforms the current scheduler used on Mira under various workload configurations.
- *MeshSched* outperforms the current scheduler used on Mira when a small portion of jobs are communication-sensitive. When a large portion of jobs are communication-sensitive (e.g., 40%), *MeshSched* reduces system fragmentation and increases system utilization at the cost of increasing job wait time and response time.

- Since not all applications are sensitive to communication bandwidth, we find that the existing scheduling design on Mira has much room for improvement. We believe the use of the new scheduling methods can improve the overall system performance.
- In general, when a small portion of communication-sensitive jobs (e.g., no more than 10%), we encourage the use of *MeshSched*; otherwise, the use of *CFCA* is a good choise.

VI. Related Work

Many researchers have investigated resource allocation strategies on supercomputers and their impact on system performance. Evans et al. studied the variability of performance on clusters and claimed that tightly allocated jobs had better performance sparse ones [24]. Kramer and Ryan found that variability introduced by different job allocation strategies can be mitigated by periodically migrating application tasks to create larger contiguous chunks [25]. Bhateke ad Kale evaluated the positive impact of locality-aware allocations on applications performance [26]. Skinner and Kramer showed that 2–3 times of improvement of MPI_Allreduce is observed by eliminating network contention from other jobs [27]. Wright et al. quantified network contention between jobs [28]. While existing studies focus mainly on performance variation caused by job interference, our work investigates application sensitivity to communication bandwidth caused by network configuration change. Moreover, our work examines a suite of parallel benchmarks and leadership applications at large scale. The results provide a foundation for the design of a communication-aware resource management system.

A number of studies have been presented to improve resources management and scheduling on large-scale systems from various aspects. In [29], Feitelson et al. provided a detailed analysis of different scheduling strategies. Zhao et al. [30] proposed network-aware caching mechanisms on large-scale systems such as IBM's Blue Gene supercomputers. Desai et al. assessed application performance degradation on shared network and studied how to improve application performance while efficiently utilizing the available torus network [31]. Pedretti et al. showed that one can use an existing large-scale parallel computer Cray XT5 to emulate the expected imbalance of future exascale systems [32]. Their results indicate that some applications experience sudden drops in performance at certain network injection bandwidth thresholds. Yang et al. [33] and Zhou et al. [34] proposed power-aware job scheduling frameworks for supercomputer systems as a 0-1 knapsack model.

To the best of our knowledge, we are among the first to systematically investigate communication awareness for resource management and job scheduling. Furthermore, we have conducted extensive trace-based simulations to quantify the benefit of communication-aware scheduling over the existing scheduling design.

VII. Conclusion

In this paper, we have presented a detailed experimental study of a suite of parallel benchmarks and applications on the Mira system at Argonne. Our results show substantial variation in sensitivity to communication bandwidth across production applications as well as microbenchmarks. Based on these benchmark efforts, we have designed two scheduling schemes, *MeshSched* and *CFCA*, for Mira by using partitions that require fewer link resources. Our experiments prove the performance benefit obtained by these new scheduling methods. While this study targets Mira, our design is generally applicable to all Blue Gene/Q systems as well as other 5D torus connected machines. This paper demonstrates how traditional scheduling processes can be extended to efficiently manage a new resource type (e.g., network).

Several avenues are open for future work. One is to build a model to predict whether a job is sensitive to communication bandwidth based on its historical data. We also plan to implement the proposed communication-aware policy into the production scheduler used on Mira. In addition, we are expanding this work with the aim of developing a smart resource management framework for better managing non-traditional resources including I/O and power consumption.

Acknowledgments

The work at Illinois Institute of Technology is supported in part by U.S. National Science Foundation grants CNS-1320125 and CCF-1422009. The FLASH software used in this work was in part developed by the DOE NNSA-ASC OASCR Flash Center at the University of Chicago. This material is based in part upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

References

[1] "Top500 supercomputing web site." [Online]. Available: http://www.top500.org

[2] "Overview of the IBM Blue Gene/P project," *IBM J. Res. Dev.*, vol. 52, no. 1/2, pp. 199–220, Jan. 2008.

[3] A. Gara, M. A. Blumrich, D. Chen, G. L.-T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, T. A. Liebsch, M. Ohmacht, B. D. Steinmacher-Burow, T. Takken, and P. Vranas, "Overview of the Blue Gene/L system architecture," *IBM J. Res. Dev.*, vol. 49, no. 2, pp. 195–212, Mar. 2005.

[4] "Managing system software for Cray XE and Cray XT systems. Cray document." [Online]. Available: http://docs.cray.com/books/S-2393-31/

[5] C. Dong, N. Eisley, P. Heidelberger, R. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. Satterfield, B. Steinmacher-Burow, and J. Parker, "The IBM Blue Gene/Q interconnection fabric," *Micro, IEEE*, vol. 32, no. 1, pp. 32–43, 2012.

[6] Y. Ajima, Y. Takagi, T. Inoue, S. Hiramoto, and T. Shimizu, "The Tofu Interconnect," in *2011 IEEE 19th Annual Symposium on High Performance Interconnects (HOTI)*, 2011, pp. 87–94.

[7] W. Tang, Z. Lan, N. Desai, D. Buettner, and Y. Yu, "Reducing fragmentation on torus-connected supercomputers," in *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, 2011, pp. 828–839.

[8] "Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program." [Online]. Available: https://www.alcf.anl.gov/incite-program

[9] "ASCR Lleadership Computing Challenge (ALCC)." [Online]. Available: http://science.energy.gov/ascr/facilities/alcc/

[10] "Cobalt resource manager." [Online]. Available: https://trac.mcs.anl.gov/projects/cobalt

[11] W. Tang, Z. Lan, N. Desai, and D. Buettner, "Fault-aware, utility-based job scheduling on Blue Gene/P systems," in *IEEE International Conference on Cluster Computing and Workshops, 2009, CLUSTER '09.*, 2009, pp. 1–10.

[12] Z. Zhou, X. Yang, Z. Lan, P. Rich, W. Tang, V. Morozov, and N. Desai, "Bandwidth-aware resource management for extreme scale systems," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC '14), poster session*, 2014.

[13] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS parallel benchmarks," Tech. Rep., 1991.

[14] P. Fischer, J. Lottes, D. Pointer, and A. Siegel, "Petascale algorithms for reactor hydrodynamics," *Journal of Physics: Conference Series*, vol. 125, no. 1, 2008.

[15] "The FLASH code." [Online]. Available: http://www.flash.uchicago.edu/site/flashcode/

[16] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo, "FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes," *The Astrophysical Journal Supplement Series*, vol. 131, no. 1, p. 273, 2000.

[17] R. T. Fisher, L. Kadanoff, D. Q. Lamb, A. Dubey, T. Plewa, A. C. Calder, F. Cattaneo, P. Constantin, I. T. Foster, M. E. Papka, S. I. Abarzhi, S. M. Asida, P. M. Rich, C. C. Glendenin, K. Antypas, D. J. Sheeler, L. B. Reid, B. Gallagher, and S. G. Needham, "Terascale turbulence computation on BG/L using the FLASH3 code," *IBM Journal of Research and Development*, vol. 52, pp. 127–136, 12/2007 2007.

[18] M. Taylor, S. Kurien, and G. Eyink, "Recovering isotropic statistics in turbulence simulations: The Kolmogorov 4/5th-law," *Physical Review E*, vol. 68, 2003.

[19] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, 1995.

[20] D. Bailey, T. Harris, W. Saphir, R. Van Der Wijngaart, A. Woo, and M. Yarrow, "The NAS parallel benchmarks 2.0," Tech. Rep., 1995.

[21] W. Tang, N. Desai, D. Buettner, and Z. Lan, "Analyzing and adjusting user runtime estimates to improve job scheduling on the Blue Gene/P," in *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, April 2010, pp. 1–11.

[22] J. Jones and B. Nitzberg, "Scheduling for parallel supercomputing: A historical perspective of achievable utilization," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, 1999, vol. 1659, pp. 1–16.

[23] Y. Xu, Z. Zhou, W. Tang, X. Zheng, J. Wang, and Z. Lan, "Balancing job performance with system performance via locality-aware scheduling on torus-connected systems," in *Cluster Computing (CLUSTER), 2014 IEEE International Conference on*, Sept 2014, pp. 140–148.

[24] J. Evans, W. Groop, and C. Hood, "Exploring the relationship between parallel application run-time and network performance in clusters," in *28th Annual IEEE International Conference on Local Computer Networks, 2003. LCN '03 Proceedings*, Oct. 2003, pp. 538–547.

[25] W. T. C. Kramer and C. Ryan, "Performance variability of highly parallel architectures," in *Proceedings of the 2003 International Conference on Computational Science: PartIII*, ser. ICCS'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 560–569.

[26] A. Bhatele and L. Kale, "Application-specific topology-aware mapping for three dimensional topologies," in *IEEE International Symposium on Parallel and Distributed Processing, 2008. IPDPS 2008*, April 2008, pp. 1–8.

[27] D. Skinner and W. Kramer, "Understanding the causes of performance variability in HPC workloads," in *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, Oct 2005, pp. 137–149.

[28] N. Wright, S. Smallen, C. Olschanowsky, J. Hayes, and A. Snavely, "Measuring and understanding variation in benchmark performance," in *DoD High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 2009*, June 2009, pp. 438–443.

[29] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, "Parallel job scheduling strategies for parallel processing," in *Proceedings of the 10th International Conference on Job Scheduling Strategies for Parallel Processing*, ser. JSSPP'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 1–16.

[30] D. Zhao, K. Qiao, and I. Raicu, "HyCache+: Towards scalable high-performance caching middleware for parallel file systems," in *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2014, pp. 267–276.

[31] N. Desai, D. Buntinas, D. Buettner, P. Balaji, and A. Chan, "Improving resource availability by relaxing network allocation constraints on Blue Gene/P," in *International Conference on Parallel Processing, 2009. ICPP '09*, 2009, pp. 333–339.

[32] K. Pedretti, R. Brightwell, D. Doerfler, K. Hemmert, and I. Laros, JamesH., "The impact of injection bandwidth performance on application scalability," in *Recent Advances in the Message Passing Interface*, 2011, vol. 6960, pp. 237–246.

[33] X. Yang, Z. Zhou, S. Wallace, Z. Lan, W. Tang, S. Coghlan, and M. E. Papka, "Integrating dynamic pricing of electricity into energy aware scheduling for HPC systems," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13. New York, NY, USA: ACM, 2013, pp. 60:1–60:11.

[34] Z. Zhou, Z. Lan, W. Tang, and N. Desai, "Reducing energy costs for IBM Blue Gene/P via power-aware job scheduling," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, 2014, pp. 96–115.